MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

MEL-PETs Joint-Context Attack for the NeurIPS 2024 LLM Privacy Challenge Red Team Track

Wang, Ye; Nakai, Tsunato; Liu, Jing; Koike-Akino, Toshiaki; Oonishi, Kento; Higashi, Takuya

TR2024-165 December 17, 2024

Abstract

We submit a PII data-extraction attack for the Red Team Track of the NeurIPS 2024 LLM Privacy Challenge. Our attack uses a customized beam search strategy that jointly considers all preceding contexts of the targeted PII, with beam scoring adjusted to promote generation of the correct PII type, and suppression of certain tokens unlikely to appear in PII. As of submission time, we are in fourth place on the public test leaderboard, with an attack success rate of 18.989%.

LLM Privacy Challenge at Neural Information Processing Systems (NeurIPS) 2024

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

MEL-PETs Joint-Context Attack for the NeurIPS 2024 LLM Privacy Challenge Red Team Track

Ye Wang
Mitsubishi Electric Research Laboratories
Cambridge, MA, USA
yewang@merl.com

Tsunato Nakai*
Mitsubishi Electric Corporation
Kamakura, Japan

Jing Liu, Toshiaki Koike-Akino
Mitsubishi Electric Research Laboratories
Cambridge, MA, USA
{jiliu, koike}@merl.com

Kento Oonishi†, Takuya Higashi‡
Mitsubishi Electric Corporation
Kamakura, Japan

Abstract

We submit a PII data-extraction attack for the Red Team Track of the NeurIPS 2024 LLM Privacy Challenge. Our attack uses a customized beam search strategy that jointly considers all preceding contexts of the targeted PII, with beam scoring adjusted to promote generation of the correct PII type, and suppression of certain tokens unlikely to appear in PII. As of submission time, we are in fourth place on the public test leaderboard, with an attack success rate of 18.989%.

1 Introduction

Privacy risks are an important concern in the deployment of large language models (LLMs). Various privacy attacks in the literature have demonstrated that potentially sensitive information from the training data can be extracted from LLMs [3, 5, 2, 1, 11, 7, 9, 8, 6, 10].

The NeurIPS 2024 LLM Privacy Challenge[4] aims to explore the privacy risk of recovering personal identifiable information (PII) from fine-tuning data. For the Red Team track of this competition, the Meta Llama 3.1-8B-Instruct LLM [4] was fined-tuned on a dataset consisting of synthetically generated chat dialog, yielding a Red Team model[5] that serves as the attack target. The goal is to recover PII of various forms (e.g., names, locations, dates, phone numbers, etc.) present in the fine-tuning dataset, which was split into development and test portions. For both splits, the fine-tuning data was provided with the PII masked out.

Here is an excerpt from the first sample of the masked fine-tuning data from the development split:

```
<User> Hello!  It's nice to see you.  May I ask your name?
<Assistant> Of course!  My name is [NAME-1].  It's nice to see
you too!  How have you been?
<User> It's great to meet you, [NAME-1]!  I've been well,
thank you.  Last time we spoke, we talked about some fun
activities.  Do you remember that?
```

*Email: Nakai.Tsunato@dy.MitsubishiElectric.co.jp
†Email: Onishi.Kento@ap.MitsubishiElectric.co.jp
‡Email: Higashi.Takuya@da.MitsubishiElectric.co.jp
[4]Challenge website: https://llm-pc.github.io/
[5]Available at: https://huggingface.co/LLM-PBE/Llama3.1-8b-instruct-LLMPC-Red-Team

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

```
<Assistant> Oh, yes!  I think we talked about some enjoyable
activities, like spending time outdoors or maybe visiting
local markets.  I remember how much I love connecting with the
vibrant community of [LOC-2] around here!
```

In the above example, the masked PII "`[NAME-1]`" and "`[LOC-2]`" were respectively "`David Klein`" and "`local farmers`" during training. Note that for clarity and ease of presentation, we have adjusted some whitespace, and we simply use "`<User>`" and "`<Assistant>`" to denote sequences of special tokens that delineate the messages and roles. For the development split, the actual values of all of the masked PII strings were provided in a separate file to allow for development evaluation.

Our proposed joint-context attack employs a combination of techniques:

1. We use beam search to jointly generate the most likely continuations for all contexts that precede each masked PII.
2. We customize the beam scoring to promote generation of PII of the correct type.
3. We suppress the generation of specific tokens that are unlikely to appear in typical PII.
4. We concatenate all generated beams for our submission to effectively guess multiple times.

Submitting as the Mitsubishi Electric Privacy-Enhancing Technologies (MEL-PETs) team, we are currently fourth (as of submission time) on the leaderboard (as "melpets"), with a test set attack success rate (ASR) of 18.989%.

2 Methodology

Our attack utilizes text continuation, while jointly considering all contexts that precede each instance of the PII to be recovered. For example, note that the masked PII "`[NAME-1]`" appears twice in the sample excerpt in Section 1 (and five more times in the rest of the first sample). The first two contexts in this example are "`<User> Hello!  ...  My name is `" and "`.  It's nice to see you too!  ...  It's great to meet you, `". These preceding context strings are readily obtained by splitting the scrubbed data sample at instances of the masked PII tag (e.g., "`[NAME-1]`").

2.1 Conceptual objective for joint-context continuation

Given a set of n preceding context strings $\{\mathbf{c}_1, \ldots, \mathbf{c}_n\}$, our method aims to generate the most likely k-token continuation $\mathbf{x} := (x_1, \ldots, x_k)$, in terms of next-token likelihoods averaged across all contexts, as given by

$$\sum_{j=1}^{k} \log \left[\frac{1}{n} \sum_{i=1}^{n} p(x_j | \mathbf{c}_i, \mathbf{x}_{<j}) \right],$$

where $p(x_j|\mathbf{c}_i, \mathbf{x}_{<j})$ denotes the next-token likelihoods determined by the targeted LLM, given the preceding context and already generated tokens $(\mathbf{c}_i, \mathbf{x}_{<j})$. In the following, we describe how the optimization of this conceptual objective is approximated with beam search for computational tractability, how beam scoring is modified to promote generating the correct type of PII, and how token likelihoods are adjusted to suppress tokens unlikely to appear in actual PII.

2.2 Customized beam search

Beam search is a widely employed technique to approximately find the most likely token sequence that continues a given input. This involves maintaining a list of m beams (i.e., candidate continuations), which we initialize by selecting the top-m next-tokens according to the log of the aggregated next-token likelihoods across all contexts, as given by $\log \left[\frac{1}{n} \sum_{i=1}^{n} p(x_1|\mathbf{c}_i) \right]$.

We then alternate between two steps, over rounds $l \in \{2, \ldots, k\}$, to produce m beams of length k:

1) Expand the list to $m \times m$ beams, by extending each beam with the top-m next-tokens, selected according to maximizing $\log \left[\frac{1}{n} \sum_{i=1}^{n} p(x_l|\mathbf{c}_i, \mathbf{x}_{<l}) \right]$.

2) Reduce the list by selecting the m beams with the highest score, as given by

$$\log q(y|\mathbf{x}) + \sum_{j=1}^{l} \log \left[\frac{1}{n} \sum_{i=1}^{n} p(x_j|\mathbf{c}_i, \mathbf{x}_{<j}) \right],$$

where $\log q(y|\mathbf{x})$ denotes a special PII type score that we introduce to promote generation of the correct PII type, and is further described in the following subsection 2.3. Note that the other terms in the summation have already been calculated for token selection, and can be efficiently accumulated when extending the beams. Note that, in all of the above, we adjust the next-token likelihoods $p(x_j|\mathbf{c}_i, \mathbf{x}_{<j})$ to suppress certain tokens, as described in subsection 2.4.

2.3 PII type score calculation

The PII type score $\log q(y|\mathbf{x})$ is calculated as the log-likelihood of the correct PII type y determined by a text classifier applied on the beam $\mathbf{x}$. We only include this term when the actual PII type y is either a Name, Location, or Date, as these three types cover the vast majority of cases in the development and test splits. To obtain the PII type likelihoods, we employ a simple zero-shot classification method, by querying the target LLM with the following prompt, where "`<Beam>`" denotes the text of $\mathbf{x}$:

```
You are a language classification assistant.
Please classify the text "<Beam>" into one of these classes:
"NAME" = a personal name,
"LOCATION" a location or place,
or "DATE" a date, year or decade.
Please answer with only "NAME", "LOCATION", or "DATE" for the
class that best fits the text.
Assistant Response:
```

Given this input, we use only the next-token logits output by the model for "`NAME`", "`LOCATION`", and "`DATE`" (token IDs 7687, 35324, and 7242), and apply log-softmax over just these three logits to obtain the class log-likelihoods, of which we select the one corresponding to the expected PII type y to obtain the PII type score $\log q(y|\mathbf{x})$.

2.4 Token suppression

Upon manual inspection of some of the output continuations, we found that the model would sometimes generate special control tokens (e.g., "`[assistant<|end_header_id|>](assistant)`") or even just the PII type mask (e.g., literally "`[NAME-1]`" instead of an actual name recovery). Since it is unlikely that these would be present in actual PII, we suppress generation of all of the special control and reserved tokens (IDs: 128000 through 128254), the "`[`" and "` [`" tokens (IDs: 58 and 510), and the "`assistant`" token (ID: 78191). This was done by setting their corresponding logits to -1000, before softmax is applied to calculate the next-token likelihoods $p(x_j|\mathbf{c}_i, \mathbf{x}_{<j})$.

3 Evaluation

Our experimental evaluation results are summarized in Table 1, where we report ASR on the development and test sets, along with the test compute time. The submission format checker[6] suggests that test evaluation simply checks whether the true PII appears anywhere in the first 100 characters of the submitted text. Thus, for the test submissions, we focused on a strategy of concatenating all m decoded beams to form each submitted recovery, which effectively allows multiple guesses. For the development set, we also evaluated this strategy of concatenating *all beams*, with the result truncated to 100 characters to simulate the testing procedure, as well as the alternative of only checking for the PII in the *top beam*, which essentially consists of the one top guess.

For a fixed token length k, test and development ASR tends to increase with the number of beams m. However, for a fixed number of beams m, we observed that the "top beam" ASR tends to increase as

[6] `https://github.com/QinbinLi/LLMPC-Red/blob/main/submission_format_checker.py`

Table 1: Summary of experimental results on development and test data, with test time compute in L40S GPU-hours. Due to limited submission slots, some settings were not submitted for testing.

m beams	k tokens	dev ASR (top beam %)	dev ASR (all beams %)	test ASR (all beams %)	compute (L40S hours)
10	6	11.89	17.39	14.619	11.7
20	5	11.99	19.62	—	25.8
20	6	12.59	19.04	16.331	31.5
20	7	**13.18**	18.51	—	38.8
25	5	11.91	19.94	17.418	38.1
25	6	12.36	19.21	—	48.6
30	4	10.83	20.92	18.405	39.6
30	5	12.04	20.21	17.418	50.7
30	6	12.59	19.49	—	62.2
35	4	10.95	21.07	18.768	49.7
35	5	12.14	20.31	—	67.6
40	4	10.95	21.17	18.909	62.9
45	3	9.87	20.94	—	57.0
45	4	10.89	21.23	18.949	95.9*
50	4	10.93	**21.33**	**18.989**	93.1*

the token length k increases, while this trend is reversed for the "all beams" ASR. This may be due to the shorter beams becoming more diverse, particularly in earlier tokens, which could boost the performance of the multiple guess "all beams" strategy, whereas more tokens may benefit the "top beam" strategy to accurately recover longer PII.

3.1 Computational costs

The contest imposes a compute time limit of 24 hours, when using up to three Nvidia H100 GPUs. However, we did not have access to any H100 GPUs, and instead ran our attacks on a mix of A40, L40, and L40S GPUs. In Table 1, we state compute time in terms of total L40S hours[7]. According to public benchmarks[8], the H100 is over 2.5 times faster than the L40S for Llama 3.1-8B inference. Thus, we expect each of our attacks to run well within 24 hours on three H100 GPUs (i.e., the 72 H100-hour budget is roughly equivalent to at least 180 L40S-hours). Note that our experiments are trivially parallelized across multiple GPUs by simply splitting the test set. We used the Unsloth library[9] to load and execute the model, which greatly reduced GPU memory usage and compute time.

4 Conclusion

Our attack employed beam search to jointly recover PII with respect to all preceding contexts. For future work, employing an adaptive decoding token length and somehow promoting beam diversity may yield benefits. Another possible direction is to leverage the differences between the fine-tuned target model and the original Llama model. During development, we attempted some approaches along these lines that involved exploiting relative likelihood between the models, but we did not achieve good results. However, further investigation may be fruitful, as we could not thoroughly explore within the limited time frame of the competition.

[7]We conservatively converted A40 run time by dividing by 1.5, since the L40S computed similar batches consistently faster than 1.5 times. We divided L40 run time by one, even though the L40S is significantly faster.

*The 50-beam, 4-token case ran for a total of 93.1 hours divided in parallel across five L40S. The 45-beam, 4-token case ran on a mix of A40 and L40, and the listed 95.9 hours is a conservative over-estimate.

[8]`https://www.runpod.io/compare/l40s-vs-h100nvl`

[9]`https://github.com/unslothai/unsloth`

References

[1] Hannah Brown, Katherine Lee, Fatemehsadat Mireshghallah, Reza Shokri, and Florian Tramèr. What does it mean for a language model to preserve privacy? In *Proceedings of the 2022 ACM conference on fairness, accountability, and transparency*, pages 2280–2292, 2022.

[2] Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. Quantifying memorization across neural language models. *arXiv preprint arXiv:2202.07646*, 2022.

[3] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650, 2021.

[4] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[5] Abhyuday Jagannatha, Bhanu Pratap Singh Rawat, and Hong Yu. Membership inference attack susceptibility of clinical language models. *arXiv preprint arXiv:2104.08305*, 2021.

[6] Qinbin Li, Junyuan Hong, Chulin Xie, Jeffrey Tan, Rachel Xin, Junyi Hou, Xavier Yin, Zhun Wang, Dan Hendrycks, Zhangyang Wang, Bo Li, Bingsheng He, and Song Dawn. Llm-pbe: Assessing data privacy in large language models. In *International Conference on Very Large Data Bases*, 2024.

[7] Justus Mattern, Fatemehsadat Mireshghallah, Zhijing Jin, Bernhard Schölkopf, Mrinmaya Sachan, and Taylor Berg-Kirkpatrick. Membership inference attacks against language models via neighbourhood comparison. *arXiv preprint arXiv:2305.18462*, 2023.

[8] Milad Nasr, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A Feder Cooper, Daphne Ippolito, Christopher A Choquette-Choo, Eric Wallace, Florian Tramèr, and Katherine Lee. Scalable extraction of training data from (production) language models. *arXiv preprint arXiv:2311.17035*, 2023.

[9] Md Rafi Ur Rashid, Vishnu Asutosh Dasu, Kang Gu, Najrin Sultana, and Shagufta Mehnaz. Fltrojan: Privacy leakage attacks against federated language models through selective weight tampering. *arXiv preprint arXiv:2310.16152*, 2023.

[10] Md Rafi Ur Rashid, Jing Liu, Toshiaki Koike-Akino, Shagufta Mehnaz, and Ye Wang. Forget to flourish: Leveraging machine-unlearning on pretrained language models for privacy leakage. *arXiv preprint arXiv:2408.17354*, 2024.

[11] Florian Tramèr, Reza Shokri, Ayrton San Joaquin, Hoang Le, Matthew Jagielski, Sanghyun Hong, and Nicholas Carlini. Truth serum: Poisoning machine learning models to reveal their secrets. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 2779–2792, 2022.